

Features and Aggregators for Web-scale Entity Search

Uma Sawant[†]Soumen Chakrabarti[†]

ABSTRACT

We focus on two research issues in entity search: how to score a document or snippet that potentially supports a candidate entity, and how to aggregate or combine scores from different snippets into an entity score. Proximity scoring has been studied in IR outside the scope of entity search. However, aggregation has been hardwired except in a few cases where probabilistic language models are used. We instead explore simple, robust, discriminative ranking algorithms, with informative snippet features and broad families of aggregation functions. Our first contribution is a study of proximity-cognizant snippet features. In contrast with prior work which uses hardwired “proximity kernels” that implement a fixed decay with distance, we present a “universal” feature encoding which jointly expresses the perplexity (informativeness) of a query term match and the proximity of the match to the entity mention. Our second contribution is a study of aggregation functions. Rather than train the ranking algorithm on snippets and then aggregate scores, we directly train on entities such that the ranking algorithm takes into account the aggregation function being used. Our third contribution is an extensive Web-scale evaluation of the above algorithms on two data sets having quite different properties and behavior. The first one is the W3C dataset used in TREC-scale enterprise search, with pre-annotated entity mentions. The second is a Web-scale open-domain entity search dataset consisting of 500 million Web pages, which contain about 8 billion token spans annotated automatically with two million entities from 200,000 entity types in Wikipedia. On the TREC dataset, the performance of our system is comparable to the currently prevalent systems by Balog *et al.* (using Boolean associations) and MacDonald *et al.*. On the much larger and noisier Web dataset, our system delivers significantly better performance than all other systems, with 8% MAP improvement over the closest competitor.

1. INTRODUCTION

In its simplest form, entity search queries provide a type (e.g., *scientist*) and ask for entities that belong to that type and satisfy other properties, expressed through keywords (*played violin*). Entity search is a prime example of searching the “Web of Objects”¹ or going from “strings to things”² pursued currently by all major search engines.

Machine learning in general, and learning to rank (L2R) [24] in particular, can be brought to bear on entity search in two key interrelated issues:

[†]IIT Bombay; contact soumen@cse.iitb.ac.in

¹<http://research.yahoo.com/news/3440>

²<http://googleblog.blogspot.in/2012/05/introducing-knowledge-graph-things-not.html>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

- How should a context be scored wrt the query? Specifically, what form of scoring function will take into account the perplexity (rarity) of query words and their proximity to (mentions of a) candidate entity in a general, trainable fashion?
- How should evidence from many contexts be aggregated into a score or rank for an entity they support? Can the context scoring model be learnt without context labels by directly optimizing for entity scores or ranks?

Several existing formulations tackle the two issues separately.

1.1 Proximity scoring

Scoring documents and passages taking query word match proximity into account is well established [6, 30, 25] in IR, but largely outside the domain of entity search. Some systems [11, 22] use hardwired proximity scoring for entity search, without using L2R. Recently, “proximity kernels” have been used [28] in entity search based on generative language models, with tunable width parameters. However, we know of no end-to-end L2R system where the proximity scoring function is itself learnt from entity relevance judgments. As we shall see here, the issue of robust, trainable proximity scoring is far from closed.

1.2 Evidence aggregation

With very few exceptions [15, 27], entity and expert search algorithms in the IR community are heavily biased toward generative language models [12, 1, 14, 2]. In contrast, some of the best-known L2R algorithms use discriminative max-margin techniques [17, 18, 16, 4, 5, 9, 34] or conditional probability formulations [7, 32, 31]. In Web search, the best L2R algorithms are believed to perform considerably better than hardwired scoring functions from early IR systems. And yet, entity and expert search have benefited little from L2R techniques.

A likely reason is the following gap in the respective models. In learning to rank (L2R), each item to be ranked is represented by one feature vector. In entity search, each item is an entity, potentially supported by many *contexts*, which may be short token sequences or entire documents. Each context, not entity, is associated with a feature vector. On the other hand, it is far easier to get entity relevance judgments than context relevance judgments.

Owing to distributional assumptions, probabilistic retrieval models [12, 1, 14, 28, 15] hardwire the manner in which individual context scores contribute to the score (thereby rank) of an entity. As we shall see in Section 2, these forms of aggregations have certain limitation. In later work, Balog *et al.* [2] allowed non-probabilistic aggregations. Macdonald *et al.* [26, 27] were the first to systematically explore a family of aggregation functions and use them as features in a L2R setting. They also used hand-crafted rank cutoffs to eliminate noisy or unreliable support contexts. Cummins *et al.* [13] used a genetic algorithm to find a soft rank cutoff.

1.3 Our contributions

We started with the goal of unifying hitherto unconnected

work on L2R, proximity scoring and evidence aggregation into a simple and uniform learning framework. It turned out that the new framework is also more robust across diverse data sets, matching or beating all known systems.

In Section 3 we explore feature design. In contrast with earlier proximity kernel [28, 25] approaches that combine a generative language model with a decay function having tuned width parameters, we propose a very general framework for feature design that encodes information about the rarity (also called “perplexity”, often measured via inverse document frequency) of query words matched in a context, as well as their distance from the candidate entity mention. In particular, we do not combine these two signals in a hard-wired manner.

In Section 4 we explore trainable evidence aggregation. In past work, only Fang *et al.* [15] proposed a document scoring model that was trained using end-to-end entity relevance judgment. We propose a family of pairwise ranking loss [24] optimization problems to deal uniformly with a variety of context score aggregation functions.

In Section 5 we present a detailed experimental study of the above approaches using two data sets. The first one is W3C dataset, from TREC expert search task used in many earlier papers. This corpus has under 350,000 documents from the W3C Web site with six different types of web pages (emails, code, wiki, personal homepages, web and misc). Since the dataset was used for enterprise search track, there is only one entity type: *person*. We performed no special processing for specific types of pages. The query set for this dataset contains 50 and 49 “topics” from the TREC 2005 and 2006 enterprise tracks. Relevance judgements were also provided, with about 4400 relevant candidates for the 99 queries. To facilitate standardization, we used the annotated version of W3C dataset prepared by Jianhan Zhu, available from <https://ir.nist.gov/w3c/contrib/W3Ctagged.html>, containing about 1.6 million annotations.

The second corpus is a representative Web crawl from a commercial search engine, with 500 million spam-free English documents. Token spans that are likely entity mentions are annotated in advance with IDs from among two million entities belonging to over 200,000 types from YAGO [29]. These annotations (about 8 billion) are then indexed along with text. We use 845 entity search queries collected from many years of TREC and INEX competitions, leading to 93 million contexts supporting candidate entities. This is perhaps among the first Web scale entity ranking testbeds where *all* candidate contexts can be analyzed without depending on a black-box document-level ranking function with possibly extraneous scoring considerations like PageRank or click statistics. We will place our code and data in the public domain to promote Web-scale entity ranking research.

1.4 Results

- Purely probabilistic language models that use an *expectation* over contexts lose vital signal in $|S_e|$, the number of contexts supporting candidate e .
- However, perplexity+proximity features add further statistically significant accuracy to just context count. Very simple features that encode perplexity (rarity) of query term matches and their proximity from the entity mention are better than fitting proximity kernels.
- On TREC, a simple non-probabilistic sum-of-context-score scheme [2, model 2, Boolean association] and a

voting scheme [27, 26] are competitive. However, our system gives comparable performance.

- On the Web testbed, our system is statistically significantly superior to all prior systems. Thus, the two data sets behave differently. Our system is more robust to the larger corpus with noisy entity recognition.

2. RELATED WORK

We set up some uniform notation. A query is denoted q . Here we will model q as a set of words and possibly phrases. Some of these may be compulsory for a match, others are optional. The set of candidate entities for q is denoted E_q , dropping the subscript if unnecessary. $e \in E_q$ is a candidate entity (in earlier work sometimes named c or ca).

A context supporting a candidate entity may be a whole document or a short span of tokens (which we call a *snippet*) approximately centered on a mention of the entity. An entity may be mentioned in multiple places in a document. Likewise, a query term may appear several times in a document, or even in a snippet. In this section, we will use S_e to denote the set of contexts that potentially support e , without committing on whether x is a document or snippet. $x \in S_e$ is one context.

The dominant language modeling approaches find the score of context x as $\prod_{t \in q} \Pr(t|x, e)$, and then aggregate these somehow over x to find a score for e .

2.1 Scoring one supporting context

Early expert search systems [1, 26] did not use proximity signals, and instead scored the whole supporting document. Proximity scoring outside expert search began around the same time [6, 30] or later [25]. Petkova *et al.* [28] first used proximity scoring in expert search using “kernels”. A proximity kernel $k(i, o)$ is a non-negative function of a term offset i and an entity mention offset o , that decreases with $|i - o|$. Instead of using terms from document x uniformly to construct a language model $\Pr(t|\theta_x)$, they use k to construct a position-sensitive language model $\Pr(t|\theta_{x,o})$ where the contribution of the term t_i at offset i is scaled by $k(i, o)$. Ranking accuracy is not very sensitive to the form of k ; a Gaussian centered at o works well.

Proximity kernels in generative language models. Note that, by definition, $\sum_t \Pr(t|\theta_{x,o}) = 1$. Consider entities e_1, e_2 supported by documents x_1, x_2 . In x_1 , e_1 is mentioned at $o_1 = 10$, e_2 in x_2 at $o_2 = 100$. Say there are two query terms, and they occur at positions 8, 13 in x_1 and 80, 130 in x_2 . Then the models $\Pr(t|\theta_{x,o})$ will be identical for $x = x_1, x_2$, and the absolute proximity information will be lost. Based on only x_1, x_2 , there would be no reason to prefer e_1 over e_2 . This is an important limitation of proximity-based language models that has not been highlighted before, and that warrants an investigation of purely feature-based approaches, that we present in Section 3.

2.2 Aggregating noisy evidence

Balog *et al.* [1, 2] were among the first to popularize generative language models, originally used in traditional IR [20], to expert search. Their best model (which we call Balog2) proceeds as $\Pr(q|e) = \sum_{x \in S_e} \Pr(q|x, e) \Pr(x|e)$. This leads to a **sum-product** form:

$$\Pr(q|e) = \sum_{x \in S_e} \left(\prod_{t \in q} \Pr(t|x, e) \right) \Pr(x|e). \quad (\text{SumProd})$$

The event space associated with $\Pr(x|e)$ has been somewhat murky; in particular, if an estimate is used such that $\sum_x \Pr(x|e) = 1$, (SumProd) effectively becomes a weighted average or expectation over support documents.

Later, Balog *et al.* [2] proposed a non-probabilistic scoring scheme by assuming uniform priors over documents and entities:

$$\begin{aligned} \Pr(q|e) &\approx \Pr(q|x) \frac{\Pr(e|x) \Pr(x)}{\Pr(e)} \\ &= \frac{1}{|S_e|} \frac{1}{|E_q|} \sum_{x \in S_e} \Pr(q|x) \Pr(e|x). \end{aligned}$$

and then simply omitting the division by $|S_e|$, effectively just adding up context scores, instead of averaging them. This retains the signal in the absolute support $|S_e|$, also highlighted as vital by others [26]. (Note that $|E_q|$, the number of candidate entities for query q , can be ignored even in a truly probabilistic framework, as it is fixed for the query.)

Macdonald and Ounis [26] provided among the first systematic studies of a space of possible aggregation functions in collecting evidence from contexts. However, the paradigm was restricted to first computing a (fixed, not learnt) score for each context, lining up a number of aggregates (such as min, max, sum, average, etc.) and then learning a linear combination among these. They did not unify voting with feature-based proximity scoring.

Curiously, Macdonald and Ounis [26] found that the “ExpCombMNZ” aggregate feature, defined as

$$|S_e| \sum_{x \in S_e} \exp(\text{score}(x, q)), \quad (\text{ExpCombMNZ})$$

consistently performed best. Here *score* is any function used for calculating match for document x w.r.t query q . Standard examples of such functions include BM25 [19] and TFIDF cosine. This is much more extreme than Balog’s sum: large scores, exponentiated, will overwhelm smaller scores, and, instead of dividing by $|S_e|$, we *multiply*. This effect can also be achieved by a rank cutoff [13, 27] or a soft-OR aggregation [22].

2.3 L2R based on entity relevance

Fang *et al.* [15] propose a noteworthy exception to the above paradigm: write

$$\Pr(e|q) = \sum_{x \in S_e} \frac{\Pr(x)}{\Pr(e)} \Pr(R_{q,x} = 1|q, x) \Pr(R_{x,e} = 1|x, e) \quad (1)$$

with two hidden Boolean random variables $R_{q,x}, R_{x,e}$. Now model each component $\Pr(R_{q,x} = 1|q, x)$ and $\Pr(R_{x,e} = 1|x, e)$ as a logistic regression. The formulation is nice in that it permits training from labeled entities alone; no labeling of contexts is needed. However, this flexibility results in a non-convex learning problem. Also, thanks to the $\Pr(x)$ term, the signal in $|S_e|$ is still lost. Their loss function is itemwise, not pairwise or listwise [24]. (In contrast, ours is pairwise like RANKSVM [18].) Furthermore, Fang *et al.* have no mechanism to capture proximity through features.

2.4 Some other related systems

Some systems for large-scale entity search [11, 22, 10] have been reported in the database community. Reminiscent of

Macdonald, Cummins and coauthors, ENTITYRANK [11] assumes additive aggregation of the form $\sum_x p(x) \text{score}(e, x, q)$ where x is a page and $p(x)$ its PageRank. Proximity scoring was hardwired. No learning was involved. EntityEngine [22] is the only system to have use a soft-or aggregation, but no feature-based learning was involved. None of these systems supported open-domain entities; the largest number of broad entity types supported was 21 [10].

2.5 Overview of our unified framework

The above picture is somewhat diverse and chaotic, and our main goal is to unify all the above efforts in a uniform, trainable feature-based discriminative ranking framework.

A context $x \in S_e$ has an associated (query dependent) feature vector $f_q(x, e) \in \mathbb{R}^M$. q is dropped if clear from context. Note that, in general, e is also an input to f_q . E.g., we may find that features for *people* should be different from features for *places*. Or we may use various collective statistics from S_e inside f_q . To keep learning simple, we will assume the raw score of a context is $w \cdot f_q(x, e)$ where $w \in \mathbb{R}^M$ is the context-level (proximity-cognizant) scoring model to be trained. Next we must aggregate the raw context scores into a score for e :

$$V(e) = \bigoplus \{T(w \cdot f_q(x, e)) : x \in S_e\}, \quad (\text{Aggr})$$

where \bigoplus is a suitable score aggregation operator. Entities will be sorted by decreasing $V(e)$ and presented to the user. (Aggr) is shown pictorially in Figure 1. Here $T \in \mathbb{R} \rightarrow \mathbb{R}$ is a (usually monotone) transformation such as $T(a) = a$, $T(a) = \log(1 + a)$, or $T(a) = e^a$. If T is convex and fast-growing, we get a soft-max effect, whereas if it is concave (diminishing returns) we get a soft-count effect. For some scoring schemes, $|S_e|$ may be recovered simply by using $\text{sum } T(a) = \llbracket a > 0 \rrbracket$.

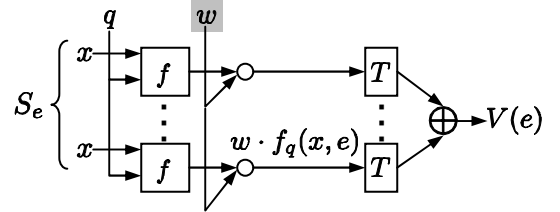


Figure 1: Feature and aggregation formula.

Most existing systems can be expressed within the above paradigm (perhaps with the training part replaced by hand tuning). E.g., in case of some language models, $w \cdot f_q(x, e)$ can be interpreted as $w_e \cdot f_q(x)$ where w_e encodes a language model for e and $f_q(x)$ selects count and position of query words in context x .

3. PROXIMITY FEATURES

In this section we design $f_q(x, e)$ to reward proximity in a trainable manner. We will assume the raw score of a context is $w \cdot f_q(x, e)$ where w is the context-level scoring model to be trained. Usually, $f_q(x, e) \geq \vec{0}$. To the objective functions that we seek to minimize, we will also add a standard regularization [3] term of the form $\frac{w \cdot w}{2\lambda^2}$, where λ is a hyperparameter that we fit via cross validation.

In what follows, the inverse document frequency or IDF(t) of a term t is defined as the inverse of the fraction of documents where the term occurs. (Instead of the inverse we

also tried the negative log [33] but results were not distinguishable.) This can also be interpreted as the “surprise value” or *perplexity* of finding t in a context. Let $IDF(q) = \sum_{t \in q} IDF(t)$.

3.1 Document vs. snippet

Evidence from different mentions of an entity in a document are scarcely independent, so it is common [25] to choose one mention from each document that is *most favorable* to the entity, i.e., with the largest score. Multiple occurrences of query words in the context offer a similar issue. When the context is a short snippet, ignoring all but the match closest to the entity mention is reported to work well [8].

3.2 Baselines

The **NoProx** baseline scores the entire document wrt the query without regard to the position/s of entity mention/s. TFIDF cosine, BM25, TFIDF-weighted Jaccard similarity, or probabilistic language models may be used. Each such score can be one feature, and w can combine them suitably. It is also common to add a constant feature (value 1, say) which allows w to effectively count the number of support contexts in S_e .

The second baseline, which we expect to be better than the first, is to use a proximity kernel [28] with a tuned width together with a probabilistic language model. This should also approximate well other similar hardwired proximity scoring schemes [11, 22, 10]. (Note that Lv and Zhai [25], while using a positional language model, were ranking documents, not entities, so they do not specify any aggregation loc.)

3.3 Perplexity-proximity features

Consider one mention of a candidate entity in a document, together with just the closest occurrences of each query word that matches in the document. Each matched word t is characterized by two quantities: the perplexity $IDF(t)$, and the distance³ ℓ (number of tokens) between the entity mention and t . We now describe three natural ways to represent the event of t occurring at distance ℓ from the entity mention.

3.3.1 Cumulative perplexity up to distance ℓ

Suppose there are three query terms $q = \{t_1, t_2, t_3\}$, only t_2, t_3 appear in the context, at distances ℓ_2, ℓ_3 from the mention. Imagine we are plotting a graph: the x-axis is distance ℓ , and the y-axis is the sum of $IDF(t)/IDF(q)$ for all t matched within distance ℓ . The plot starts at $(0, 0)$, and jumps up at any distance where there is a match, in this example, from 0 to $IDF(t_2)/IDF(q)$ at ℓ_2 , then to $(IDF(t_2) + IDF(t_3))/IDF(q)$ at ℓ_3 . The final value is the fraction of query IDF that is matched in the context (1, if all query terms were found). This forms a normalized feature space for learning w . We call these **IdfUpTo** features.

3.3.2 Grid features

Now consider a query term t that matches at distance ℓ from the mention of candidate e . Then $IDF(t)/IDF(q) \in [0, 1]$ decides the “perplexity coordinate” of the match, whereas ℓ decides the “proximity coordinate” of the match. In Figure 2, there are two query terms that match. *Capital* has

³For simplicity we use absolute distance, but left/right can also be encoded naturally using signed distance.

lower IDF, but is closer to the candidate, *Abuja* compared to *Nigeria*, with higher IDF but farther away. ...*Abuja* officially gained its status as the **capital** of *Nigeria*...

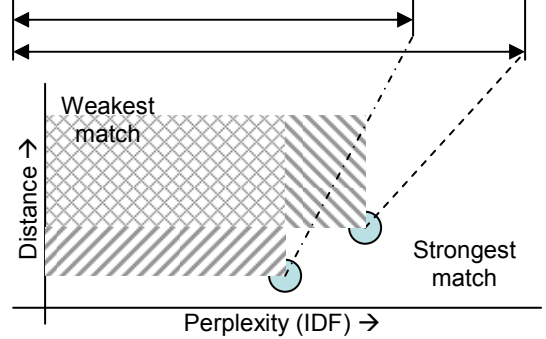


Figure 2: Perplexity-proximity grid features.

Each axis is suitably bucketed to fire one feature (i, j) in a grid of features (which is later flattened to a single index in a 1-dimensional vector $f_q(x, e)$). Every query word match results in firing one cell in the feature grid (shown as the circles). This is a “universal” encoding, without any commitment on how perplexity and proximity should be combined; the combination is decided by learning w . We call these **grid features**.

Note that w has an element $w_{i,j} \geq 0$ corresponding to each grid cell. Because our discretization is arbitrary, we do not expect $w_{i,j}$ to differ much from $w_{i \pm 1, j \pm 1}$. Therefore, this part of w should not be regularized (only) as $w_{i,j}^2/(2\lambda^2)$, but as

$$\frac{1}{2\lambda^2} \sum_{i,j} (w_{i,j} - w_{i-1,j})^2 + (w_{i,j} - w_{i,j-1})^2.$$

Assuming row $i+1$ means “more IDF” than row i and column $j+1$ means “more proximity” than column j , we may also want to enforce monotonicity constraints of the form

$$w_{i+1,j} \geq w_{i,j} \quad \text{and} \quad w_{i,j+1} \geq w_{i,j}.$$

3.3.3 Rectangle features

The above forms of constraints over the perplexity-proximity grid complicate model training, and can be avoided by a transformation of the grid features to **rectangle features**. As Figure 2 shows, each query term matched in the snippet fires one corresponding grid feature (i, j) (shown by the two circles). In the rectangle feature encoding, we also turn on all cells that have lower IDF or worse proximity. This ensures that if (i, j) and (i', j') are close together, the features fired have a large overlap (double-hatched area). Also, the farther to the south-east corner (i, j) is, the more features are fired. Note that rectangle features no longer require the above constraints, just $w_{ij} \geq 0$ is enough.

4. EVIDENCE AGGREGATION

As described in Section 2.5, we use a general expression for entity value (Aggr) that combines proximity scoring and evidence aggregation. The parameters inside (Aggr) will be trained using entity-level relevance judgment.

For query q let G_q, B_q be sets of good (relevant) and bad (irrelevant) entities. We will use g, b for good and bad entities. x_+, x_- will denote contexts potentially supporting good and bad entities. Before moving on to our suite of aggregation learners, we note that one may also attempt to

directly use L2R techniques at a context level. E.g., we can directly use RANKSVM [18] with hinge loss at the context level, to minimize wrt w the objective

$$\sum_q \frac{\sum_{\substack{g,b \\ x_+ \in S_g \\ x_- \in S_b}} \max\{0, 1 + w \cdot (f_q(x_-, b) - f_q(x_+, b))\}}{|G_q||B_q|}$$

Note that, while $|G_q||B_q|$ is used to normalize the loss across queries, the loss is not scaled down by $|S_g|$ or $|S_b|$, which would average out context support (see Section 4.2).

Context-level formulations are impractical to train. In our data set, cases of 10^{11} context pairs (x_+, x_-) are not at all rare. Even an efficient stochastic gradient or sub-gradient descent method has no hope of dealing with such scale without extreme sampling. So some form of direct aggregation to entity score is essential.

Another problem (further motivating Fang *et al.*'s work [15]) is that a context supporting a good entity is not necessarily an evidence context as judged by a human; the entity mention and some query terms may be juxtaposed coincidentally. On the other hand, acquiring context-level supervision is orders of magnitude more expensive than entity-level relevance judgment.

4.1 $|S_e|$ baseline

A baseline that is known [26] to be very competitive in our testbed is to ignore the quality of the match between query and context altogether (given at least one term overlap) and simply use the number of supporting contexts, i.e., $w = (1)$, $f_q(x, e) = (1)$, $\oplus = \sum$, and so $V(e) = |S_e|$. All other aggregations must be compared against this trivial baseline.

4.2 Sum and average

If we believe x has any signal, and all entity ranking systems believe so, we should use nontrivial $f_q(x, e)$ s. Two obvious aggregators that suggest themselves are:

$$V(e) = \sum_{x \in S_e} w \cdot f_q(x, e) \quad (\text{Sum})$$

$$\text{and } V(e) = \frac{1}{|S_e|} \sum_{x \in S_e} w \cdot f_q(x, e). \quad (\text{Avg})$$

Here $T(a) \propto a$. (Sum) mimics Balog's non-probabilistic formulation [2]. (Avg) is our approximation to the evidence aggregation done by generative language models (SumProd). Generally, for the sums above to be meaningful, we want no cancellation of terms, so we will design $f_q(\dots) \geq \vec{0}$ and constrain $w \geq \vec{0}$. (For all existing systems this is the case.)

4.3 SoftMax

Within the context of TREC entity search, Macdonald *et al.* [26], Cummins *et al.* [13] and others have noted that not all $x \in S_e$ should contribute to $V(e)$. Some of these matches are high-noise and should be tuned down (over and above a hopefully low context score itself) or eliminated. They try to achieve this effect in two different ways. Cummins *et al.* implement a soft cutoff as a weighted sum

$$V(e) = \sum_{x \in S_e} D(x; S_e) w \cdot f_q(x, e), \quad (\text{SoftCutOff})$$

where $D(x; S_e)$ is a contribution weight that may depend on, e.g., the rank of x within S_e . We present a linear program

to learn D in Section 4.7. Macdonald *et al.* instead favor high-scoring contexts by formulating

$$V(e) = \sum_{x \in S_e} T(w \cdot f_q(x, e)), \quad (\text{SoftMax})$$

where $T(\cdot)$ is a fast-growing function, such as $T(a) = e^a$. A few high scoring contexts will tend to dominate $V(e)$, hence the name. (ExpCombMNZ) is even more extreme, effectively it replaces all scores in S_e by the maximum and adds them up. We limit ourselves to $T(a) = e^a$.

4.4 SoftOr

Although experience with TREC expert search is favorable, it is not clear if/why soft-max is a universally superior choice. If a few high-quality evidence contexts should override other supporting context scores, another natural aggregator readily suggests itself: the soft-or (used in EntityEngine [22]). The premise here is

$$\Pr(e \text{ is good} | S_e) = 1 - \prod_{x \in S_e} (1 - \Pr(x \text{ is evidence}))$$

The standard technique [15] to turn $w \cdot f_q(x, e)$ into a probability is to use the sigmoid function $T(a) = \sigma(a) = 1/(1 + e^{-a})$:

$$\Pr(x \text{ is evidence for } e) = \sigma(w \cdot f_q(x, e))$$

In soft-or, \oplus is no longer \sum , and we get

$$V(e) = 1 - \prod_{x \in S_e} (1 - \sigma(w \cdot f_q(x, e))). \quad (\text{SoftOr})$$

4.5 SoftCount

In both SoftMax and SoftOr, a few large context scores can override a number of smaller context scores. An opposite policy, consistent with the observation that $|S_e|$ is a good scoring scheme by itself, is that all supporting contexts have some merit, but there is variation in their evidence quality. This suggests we use a *concave* T , with a diminishing return shape, instead of a convex one like $\exp(\cdot)$. This implements a form of soft counting. We specifically used $T(a) = \log(1 + a)$ but experience with other forms like $T(a) = a^p$ with $0 < p \leq 1$ were similar.

4.6 Training w through aggregation \oplus

The earliest L2R formulations seek to minimize the number of wrongly ordered entity pairs ("pair swaps") $g \in G_q, b \in B_q$ such that $V(b) > V(g)$. The number of pair swaps is directly related to the area under the curve (AUC) measure in machine learning, and is also related to MAP by one-sided bounds [18]. Minimizing pair swaps [17, 18] is a simple and robust L2R approach that remains hard to beat. RANK-SVM [18] proposed to train w by minimizing wrt w the pair swap hinge loss

$$\sum_q \frac{1}{|G_q||B_q|} \sum_{g,b} \max\{0, 1 + V(b) - V(g)\}. \quad (\text{HingeLoss})$$

We will avoid dual solutions and use simple gradient-descent optimizers by replacing the hinge loss $\max\{0, 1 + V(b) - V(g)\}$ with the continuous and differentiable soft hinge loss

$$\text{SH}(1 + V(b) - V(g)), \text{ where } \text{SH}(a) = \log(1 + e^a).$$

Note that $\text{SH}'(a) = \frac{e^a}{1+e^a} = \sigma(a)$, the sigmoid function. The generic gradient of the above loss wrt w is

$$\sum_q \frac{1}{|G_q||B_q|} \sum_{g,b} \sigma(1 + V(b) - V(g)) \left[\frac{dV(b)}{dw} - \frac{dV(g)}{dw} \right],$$

(Gradient)

so all that remains is to plug in $V(e)$ and $dV(e)/dw$ for all the cases discussed. When $V(e) = \sum_x T(w \cdot f_q(x, e))$, this is simply $dV(e)/dw = \sum_x T'(w \cdot f_q(x, e)) f_q(x, e)$. The \oplus =(SoftOr) case is not additive, but with a little work we can derive:

$$\frac{dV(e)}{dw} = \sum_{x \in S_e} \frac{f_q(x, e)}{1 + e^{-w \cdot f_q(x, e)}} \prod_{x' \in S_e} \left(1 - \frac{1}{1 + e^{-w \cdot f_q(x, e)}} \right).$$

The soft hinge objective is a convex optimization only in the case $\oplus = \sum$ and $T(a) = a$. However, (quasi) Newton optimizers like LBFGS [23] tend to behave acceptably well even when given non-convex problems from this domain [21]. Extending our approach to using listwise ranking losses [24] is a possible direction for future work.

4.7 Training a soft cutoff by rank

If we assume that w has been trained and fixed, we can set up a simple linear program (LP) for implementing the kind of soft cutoff sought by Macdonalds *et al.* [27] and Cummins *et al.* [13]. To make it convenient to use an LP, we will revert from soft hinge to (HingeLoss). For good, bad entity pair g, b , as in RANKSVM, define slack variable $H(g, b) \geq 0$, with constraint

$$\forall g, b: \quad H(g, b) \geq 1 + V(b) - V(g);$$

the objective to minimize will be $\sum_q \frac{1}{|G_q||B_q|} \sum_{g,b} H(g, b)$.

The soft cutoff decay will be modeled using more variables $D(r)$ where r is a rank. Variables D are constrained by

$$\forall r: \quad D(r) \geq D(r+1), \quad \text{and} \quad D(r) \geq 0.$$

Now that w is fixed, all context scores are also fixed. Let the rank of context x within S_e be r_x . Then we have

$$V(e) = \sum_{x \in S_e} D(r_x)(w \cdot f_q(x, e)),$$

which can be used to express $H(g, b)$ directly in terms of $D(\cdot)$ and known constants. As in support vector machines, to limit the overfitting powers of $D(\cdot)$, we tack on to the objective a regularization term of the form $D(0)/\lambda$ where λ is a tuned width parameter in the same sense as $w \cdot w/(2\lambda^2)$ is used in SVM regularization. Summarizing, the objective will be

$$\min_{H \geq 0, D \geq 0} \frac{D(0)}{\lambda} + \sum_q \frac{1}{|G_q||B_q|} \sum_{g,b} H(g, b)$$

subject to the above constraints. Different entities will have diverse $|S_e|$. To share a decay profile $D(r)$ across these, we allocated parameters in $D(\cdot)$ for deciles of ranks. We tried many other rank bucketing approaches but they did not affect the results significantly.

5. EXPERIMENTS

5.1 Data sets and statistics

We use two data sets and tasks. The first one is the standard TREC enterprise track expert search task used in most prior work on expert/entity search. The corpus has 331,000 documents from W3C Web site. Mentions of persons (experts) have been annotated (presumably with near-perfect accuracy) throughout the corpus. The total number of annotations is 1.6 million. The only type of entity sought is a person (expert) on a given topic, so queries are just bags of words. On an average a query involves 680 candidate experts. Expert labels (relevant/irrelevant) were provided with the queries. We chose this reference corpus to make sure our implementation of reported earlier systems is faithful, with ranking accuracy scores closely matching published numbers.

But our real interest is in open-domain Web-scale entity search, in which, as we shall see, competing systems behave rather differently compared to TREC. Our second testbed uses a 500 million-page Web corpus from a commercial search engine. Token spans that are likely entity mentions are annotated ahead of time with IDs from among two million entities belonging to over 200,000 types from YAGO [29]. About eight billion resulting annotations were then indexed along with text.

The next step was to collect queries with relevance-judged entities. We used 845 queries from many years of TREC and INEX. The queries were expressed as natural language questions. There are two steps to answering these: identify the answer type from among our 200,000 types, and use (some of) the other query words to probe the text index. To isolate these two steps, and to align the task to the TREC task, we had five people rewrite the query into the two constituents: the answer type and words/phrases to be matched literally. Some examples follow:

- The original query *What is the name of the vaccine for chicken pox* was labeled as seeking an entity of the type **wordnet_vaccine_104517535** with one or more of these words matched close by: **drug + "chicken pox" + vaccine**.
- Likewise, for the original query *Rotary engines were manufactured by which company*, the type sought is **wordnet_manufacturer_108060446** and the keyword literals may be **company + rotary + engine**.

The translation was done by proficient search engine users who use + and quotes properly. This is not unfair, because the same queries and retrieval algorithms are available to all competing algorithms. The queries are available for anonymous viewing at <http://goo.gl/T2Kkp>. The five volunteers also curated positive and negative entity instances from TREC, INEX and the Web; that data will also be made available in the public domain.

On an average a query leads to evaluating 1884 candidate entities and 110231 contexts, for a total of 93 million contexts over 845 queries. Figure 3 shows the distribution of the number of candidate entities per query. Figure 4 shows, for relevant and irrelevant entities, the number of supporting contexts. Both plots show heavily skewed behavior. In particular, from Figure 4, we see that some entities are enormously more popular on the Web compared to others. Although some good entities have huge $|S_e|$, we also see that lower down, good and bad entities are well-mixed in terms of $|S_e|$, and therefore good-bad separation during ranking remains a challenging problem.

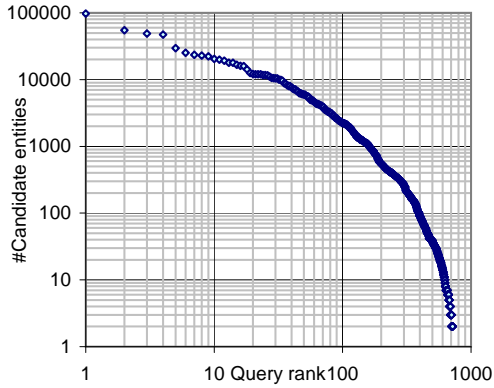


Figure 3: Candidate entities per query.

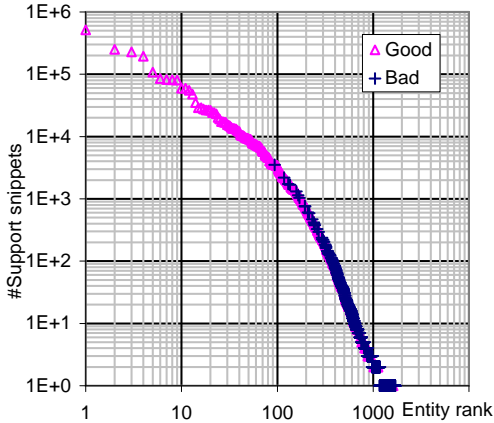


Figure 4: Distribution of supporting contexts ($|S_e|$) per candidate entity, good and bad.

Figure 5 shows context score distributions within some sampled S_e s for three good and three bad entities. All entities are fairly mixed together in the chart of context score vs. context rank. Therefore, as with $|S_e|$ in Figure 4, entities are not easy to separate on the basis of score alone. Three major steps/levels are seen, corresponding to presence or absence of query keywords. Within each broad level, smaller variations near the edges are because of diverse distances at which query term matches occur.

5.2 Measurements

Unless stated, our uniform evaluation policy was *leave one query out cross validation*. We marked each query as the test query, and trained our parameters on the remaining queries. Then we evaluated the trained parameters on the single test query. Finally we averaged accuracy measures across all test queries. This is computationally intensive, but exploits training data maximally and gives a more reliable estimate. In some cases (SoftMax and SoftOr) we reduced the computational cost of optimization using standard five fold cross validation across queries. We report entity level MAP, MRR, NDCG@5, NDCG@10, and pairs of good/bad entities that are reversed in rank. The last measure is best if small; others are best if large.

5.3 Effect of proximity features

To study the two interacting policies (features and aggregation), here we will fix the aggregation policy to our overall best (unweighted sum of context scores, see Section 5.4), and vary the design of features.

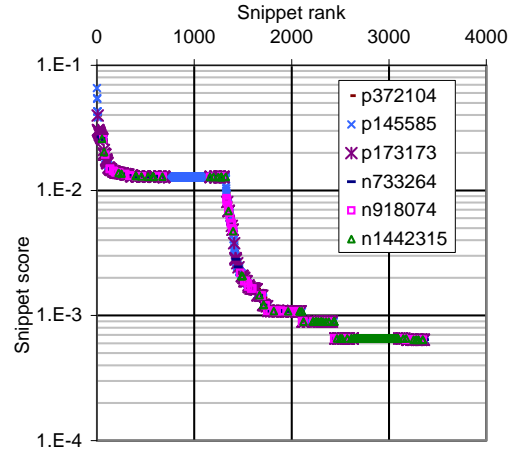


Figure 5: Distribution of context scores in a few S_e s for three good and three bad entities.

	MAP	MRR	NDCG@5	NDCG@10	Pairswap
Only $ S_e $	0.542 [↓]	0.559 [↓]	0.641 [↓]	0.660 [↓]	0.211 [↓]
NoProx	0.559	0.578 [↓]	0.661	0.675	0.203
NoProx + IdfUpto	0.560	0.581	0.661	0.672	0.221 [↓]
NoProx + rectangle	0.563	0.585	0.656	0.675	0.202

Figure 6: Proximity features compared (Web data).

Figure 6 compares various proximity features for the Web corpus. Rectangle features lead to statistically significant (paired t-test at $p = 0.05$) improvements over not using proximity signals. Here, and in all tables comparing different settings/systems, in each column, the largest value is shown in **boldface**, and other quantities in the same column that are statistically significantly smaller are suffixed with a ‘[↓]’. Proximity kernels [28, 25] perform worse than the numbers in Figure 6, but this is in part due to the way probabilistic language models are built around the kernels (also see end-to-end comparisons in Section 5.5).

Figure 6 shows that $|S_e|$ is already a strong signal for Web data, which concurs with [27]. However, the proximity features bring out significant gains beyond $|S_e|$.

Figure 7 repeats the feature comparison for TREC. A prominent observation is that $|S_e|$ is an excellent single feature for the Web, but not at all for TREC. Given that TREC-QA/INEX queries involve entities well-known to the Web, the “embarrassment of riches” represented in 500 million documents ensures that retrieved contexts are of generally high quality, so just counting them up is not too bad. In contrast, in the much smaller TREC corpus, accidental similarities between the query and the whole document bring in a large fraction of poor quality contexts. This is also confirmed by a later experiment: while the TREC task benefits from rank-based cutoffs, the Web task does not.

Figure 8 shows the contributions made to a context score by features firing in each cell shown earlier in Figure 2. If proximity or IDF had no signal, the result would be a flat-valued weight grid. Instead we see visible increase in score contributions as we go from the (low-IDF, large-distance) corner toward the (high-IDF, small-distance) corner of the

TREC 05	MAP	MRR	NDCG@5	NDCG@10	Pairsap
Only $ S_e $	0.086 [↓]	0.271 [↓]	0.310 [↓]	0.320 [↓]	0.459 [↓]
NoProx	0.172 [↓]	0.511 [↓]	0.567 [↓]	0.567 [↓]	0.371
NoProx + IdfUpto	0.187	0.521	0.585 [↓]	0.570 [↓]	0.373
NoProx + rectangle	0.188	0.523	0.606	0.606	0.370
TREC 06	MAP	MRR	NDCG@5	NDCG@10	Pairsap
Only $ S_e $	0.286 [↓]	0.728 [↓]	0.718 [↓]	0.715 [↓]	0.277 [↓]
NoProx	0.468	0.897	0.900	0.877	0.211
NoProx + IdfUpto	0.459 [↓]	0.884 [↓]	0.892	0.865 [↓]	0.222 [↓]
NoProx + rectangle	0.477	0.909	0.902	0.879	0.211

Figure 7: Proximity features compared (TREC).

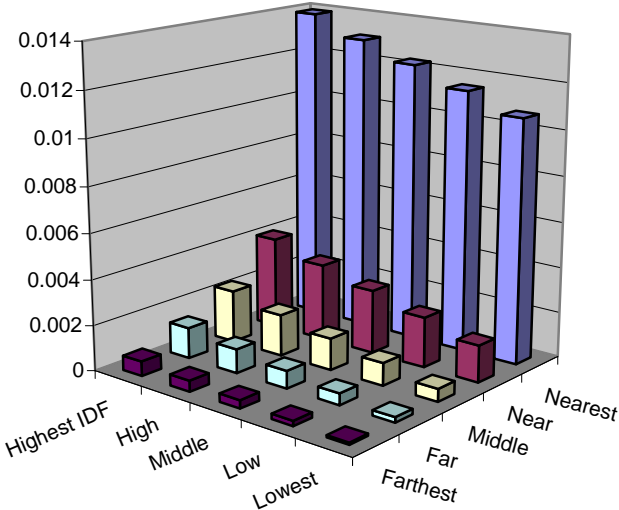


Figure 8: Sample rectangle model weights over the feature grid.

5.4 Effect of aggregation policies

In this subsection we fix the feature representation to the best reported in the previous subsection, and explore aggregation schemes. The research questions are:

- Prior work [13, 27] suggest that contexts in S_e should not contribute symmetrically to the score of e . Does SoftMax or SoftOr perform better than a simple (linear) sum of context scores?
- Can we get additional mileage beyond linear sum by making $T(\cdot)$ sublinear, i.e., using a SoftCount?
- Can we get improvements by using ranks within S_e to implement a soft cutoff (see subsection 4.7)?

Figure 9 shows (for Web data) the effect of choosing score transformer T and aggregator \oplus in various ways for Web data. Note that w is trained through this choice of T, \oplus as explained in section 4.6 and illustrated in Figure 1. The

	MAP	MRR	NDCG@5	NDCG@10	Pairsap
SoftMax	0.539 [↓]	0.557 [↓]	0.639 [↓]	0.657 [↓]	0.216
SoftOr	0.357 [↓]	0.374 [↓]	0.440 [↓]	0.460 [↓]	0.238 [↓]
SoftCutoff	0.575	0.592	0.660	0.675	0.210
Sum	0.576	0.597	0.666	0.681	0.207
Average	0.181 [↓]	0.188 [↓]	0.227 [↓]	0.256 [↓]	0.292 [↓]
SoftCount	0.554 [↓]	0.574 [↓]	0.661	0.675	0.207
$ S_e $	0.542 [↓]	0.559 [↓]	0.641 [↓]	0.660 [↓]	0.211

Figure 9: Effect of T, \oplus and soft cutoffs (Web).

top three rows show the discriminative aggregation schemes where high-scoring contexts in S_e get additional preference. SoftMax uses $\sum_x \exp(w \cdot f_q(x, e))$, and SoftOr is as described in subsection 4.4. SoftCutoff follows subsection 4.7. The fourth row shows simple sum $\sum_x w \cdot f_q(x, e)$ with all contexts treated symmetrically. The fifth uses (Avg) instead of sum, and the last two rows show sublinear aggregation (subsection 4.5) and plain $|S_e|$ as a trivial baseline. Figure 10 shows, for TREC, the counterpart of Figure 9.

Linear sum is the clear winner. It is curious that neither superlinear nor sublinear aggregation beats linear sum. This could be because linear sum gives a convex optimization while SoftMax, SoftOr and SoftCount get trapped in local optima, or because there is something fundamental about linear sum; this is worthwhile researching further. Also note that averaging, as against summing, performs poorly, and $|S_e|$ by itself is not as good as linear sum. Even when scoring was done using linear sum and the SoftCutoff linear program was used to remove low-scoring contexts' contributions to entity scores, accuracy dropped. This lends additional evidence that symmetric context contribution to entity score is the best policy.

5.5 End-to-end comparisons

Finally, we compare our system's end-to-end accuracy against other systems, for both data sets. We compare with these prior systems:

- Balog2 [2], without any proximity signal.
- Macdonald *et al.*'s formulation [27] which uses various combinations of document scoring models, voting techniques and ranking cutoffs.
- Petkova *et al.*'s formulation using proximity kernel and generative language model [28].

Although Lv and Zhai [25] used positional language models, they did so for document, not entity ranking. Therefore they did not specify the all-important aggregation logic needed to turn their system into an entity search system, and so we cannot directly compare with them.

Fang *et al.*'s formulation [15] makes (probabilistic) annotation a query-time activity along with score aggregation. While novel, this approach is not practical at Web scale, where entities may need to be annotated in millions of snippets at query time. In both our data sets, annotation is conducted offline, which effectively turns Fang *et al.*'s system into a single logistic regression for context scoring, followed by an expectation over contexts, which we already know as surpassed by $\oplus = \sum$.

Petkova *et al.* [28] not only suffer from the same weighted average limitation, but is also impractical to implement on a

TREC 05	MAP	MRR	NDCG@5	NDCG@10	Pairsap
SoftMax	0.072 [↓]	0.270 [↓]	0.288 [↓]	0.294 [↓]	0.516 [↓]
SoftOr	0.075 [↓]	0.217 [↓]	0.255 [↓]	0.299 [↓]	0.475 [↓]
SoftCutoff	0.204	0.551 [↓]	0.400 [↓]	0.610	0.362
Sum	0.207	0.598	0.650	0.614	0.369
Average	0.111 [↓]	0.311 [↓]	0.380 [↓]	0.372 [↓]	0.437 [↓]
SoftCount	0.195	0.524 [↓]	0.580 [↓]	0.590 [↓]	0.372
$ S_e $	0.086 [↓]	0.271 [↓]	0.310 [↓]	0.320 [↓]	0.459 [↓]

TREC 06	MAP	MRR	NDCG@5	NDCG@10	Pairsap
SoftMax	0.257 [↓]	0.710 [↓]	0.715 [↓]	0.684 [↓]	0.304 [↓]
SoftOr	0.214 [↓]	0.541 [↓]	0.588 [↓]	0.610 [↓]	0.319 [↓]
SoftCutoff	0.463 [↓]	0.891 [↓]	0.878 [↓]	0.861 [↓]	0.210
Sum	0.516	0.933	0.903	0.894	0.203
Average	0.148 [↓]	0.302 [↓]	0.370 [↓]	0.379 [↓]	0.364 [↓]
SoftCount	0.456 [↓]	0.912 [↓]	0.881 [↓]	0.855 [↓]	0.217 [↓]
$ S_e $	0.286 [↓]	0.728 [↓]	0.718 [↓]	0.715 [↓]	0.277 [↓]

Figure 10: Aggregation choices for TREC.

Web-scale distributed index. Instead of (SumProd), Petkova evaluates the kernel over all documents for each entity mention, then combines them. Therefore we can present numbers for TREC alone.

TREC 05	MAP	MRR	NDCG@5	NDCG@10	Pairsap
Balog2	0.214	0.604	0.623 [↓]	0.617	0.347 [↓]
Macdonald	0.258	0.605	0.620 [↓]	0.623	0.333
Petkova	0.101 [↓]	0.370 [↓]	0.371 [↓]	0.373 [↓]	0.549 [↓]
Sum	0.207 [↓]	0.598	0.650	0.614	0.369 [↓]

TREC 06	MAP	MRR	NDCG@5	NDCG@10	Pairsap
Balog2	0.528	0.925	0.923	0.921	0.190
Macdonald	0.510 [↓]	0.912 [↓]	0.917 [↓]	0.903 [↓]	0.198
Petkova	0.318 [↓]	0.731 [↓]	0.747 [↓]	0.751 [↓]	0.332 [↓]
Sum	0.516	0.933	0.903 [↓]	0.894 [↓]	0.203

Figure 11: End to end comparisons (TREC).

Figure 11 shows TREC results. For TREC 2005, Macdonald is better than Balog2. For TREC 2006, Balog2 is better than Macdonald. Our system scores slightly less, but is occasionally the best (NDCG@5 for TREC 05 and MRR for TREC 06). Petkova implements the “Balog1” model [1, equation (3)], known for higher recall and lower precision, and falls behind the others.

Figure 12 is for the Web data set. Here our system performs consistently better than all previous approaches tested, and all differences are significant. Apart from our parameter learning, Balog2 does not exploit proximity. Also, as Figure 9 (SoftCutoff row) shows, rank-based cutoffs work

	MAP	MRR	NDCG@5	NDCG@10	Pairsap
Balog2	0.452 [↓]	0.468 [↓]	0.543 [↓]	0.565 [↓]	0.173 [↓]
Macdonald	0.535 [↓]	0.553 [↓]	0.616 [↓]	0.634 [↓]	0.222 [↓]
Sum	0.576	0.597	0.666	0.681	0.207

Figure 12: End to end comparisons (Web).

worse than symmetric aggregation for the Web, which may explain why Sum beats Macdonald.

6. CONCLUSION

We presented a system that unifies diverse, unconnected approaches for context scoring and entity-level score aggregation into a simple, feature-based, trainable, discriminative and robust framework for entity ranking. We evaluated our system using two data sets. On the TREC data set, we are best or close on most evaluation criteria. On the Web data set, we are considerably ahead of the competition on all criteria. The main lessons, some confirming earlier wisdom, were:

- Simple rectangle features, that capture query match perplexity and lexical proximity, work better than proximity kernels in conjunction with probabilistic language models.
- In case of TREC, $|S_e|$ is a valuable signal; for the Web, it is not. In all cases, adding more features helped.
- How we aggregate makes or breaks algorithms. In general, we should *sum*, *not average* evidence. This has serious implications for probabilistic entity scores that look like $\sum_x (\dots) \Pr(x|e)$.
- Sublinear (SoftCount) or superlinear (SoftMax) context score combinations did not yield better ranking than a simple linear combination; neither did SoftOr. Rank-based asymmetry in score aggregation did not help for Web data.

Part of our contribution is a fully implemented search system that answers in a few seconds open-domain entity queries using two million entities and 200,000 types executed over 500 million Web pages, soon to be upgraded to two billion pages. Our code, a demo, and a search API (as a Web service) will be placed in the public domain.

7. REFERENCES

- [1] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR Conference*, pages 43–50, 2006.
- [2] K. Balog, L. Azzopardi, and M. de Rijke. A language modeling framework for expert finding. *Information Processing and Management*, 45(1):1–19, 2009.
- [3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML*, 2005.
- [5] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *NIPS Conference*, 2006.
- [6] S. Büttcher, C. L. A. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *SIGIR Conference*, pages 621–622.

- ACM, 2006.
- [7] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *ICML*, pages 129–136, 2007.
 - [8] S. Chakrabarti, K. Puniyani, and S. Das. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In *WWW Conference*, pages 717–726, Edinburgh, May 2006.
 - [9] O. Chapelle, Q. Le, and A. Smola. Large margin optimization of ranking measures. In *NIPS 2007 Workshop on Machine Learning for Web Search*, 2007.
 - [10] T. Cheng and K. C.-C. Chang. Beyond pages: supporting efficient, scalable entity search with dual-inversion index. In *EDBT*, pages 15–26. ACM, 2010.
 - [11] T. Cheng, X. Yan, and K. C. Chang. EntityRank: Searching entities directly and holistically. In *VLDB Conference*, pages 387–398, Sept. 2007.
 - [12] W. B. Croft and J. Lafferty. *Language Modeling for Information Retrieval*. Kluwer Academic Publishers, Norwell, MA, USA, 2003.
 - [13] R. Cummins, M. Lalmas, and C. O’Riordan. Learning aggregation functions for expert search. In *European Conference on Artificial Intelligence*, pages 535–540, Lisbon, 2010.
 - [14] H. Fang and C. Zhai. Probabilistic models for expert finding. In *ECIR*, pages 418–430, 2007.
 - [15] Y. Fang, L. Si, and A. P. Mathur. Discriminative models of integrating document evidence and document-candidate associations for expert search. In *SIGIR Conference*, 2010.
 - [16] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.
 - [17] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *International Conference on Artificial Neural Networks*, pages 97–102, 1999.
 - [18] T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD Conference*, pages 133–142. ACM, 2002.
 - [19] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments (parts 1 and 2). *Information Processing and Management*, 36(6):779–840, 2000.
 - [20] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR Conference*, New Orleans, 2001.
 - [21] D. Li and M. Fukushima. On the global convergence of BFGS method for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 11:11–4, 1999.
 - [22] X. Li, C. Li, and C. Yu. EntityEngine: Answering entity-relationship queries using shallow semantics. In *CIKM*, Oct. 2010. (demo).
 - [23] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)):503–528, 1989.
 - [24] T.-Y. Liu. Learning to rank for information retrieval. In *Foundations and Trends in Information Retrieval*, volume 3, pages 225–331. Now Publishers, 2009.
 - [25] Y. Lv and C. Zhai. Positional language models for information retrieval. In *SIGIR Conference*, pages 299–306, 2009.
 - [26] C. Macdonald and I. Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *CIKM*, pages 387–396, 2006.
 - [27] C. Macdonald and I. Ounis. Learning models for ranking aggregates. In *Advances in Information Retrieval*, volume 6611 of *LNCIS*, pages 517–529. 2011.
 - [28] D. Petkova and W. B. Croft. Proximity-based document representation for named entity retrieval. In *CIKM*, pages 731–740. ACM, 2007.
 - [29] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge unifying WordNet and Wikipedia. In *WWW Conference*, pages 697–706. ACM Press, 2007.
 - [30] T. Tao and C. Zhai. An exploration of proximity measures in information retrieval. In *SIGIR Conference*, pages 295–302. ACM, 2007.
 - [31] H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to rank by optimizing NDCG measure. In *NIPS Conference*, pages 1883–1891, 2009.
 - [32] M. N. Volkovs and R. S. Zemel. BoltzRank: Learning to maximize expected ranking gain. In *ICML*, 2009.
 - [33] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan-Kaufmann, May 1999.
 - [34] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR Conference*, pages 271–278, 2007.